Yahoo! UI Library

YAHOO Global 2.9.0

Yahoo! UI Library > yahoo > YAHOO.js (source view) Search:

Modules

- animation
- autocomplete
- button
- calendar
- carousel
- charts
- colorpicker
- connection
- container
- cookie
- datasource
- datatable
- datemath
- dom
- dragdrop
- editor
- element
- element-delegate
- event
- event-delegate
- event-mouseenter
- event-simulate
- get
- history
- imagecropper
- imageloader
- json
- layout
- logger
- menu
- paginator
- profiler
- profilerviewer
- progressbar
- resize
- selector
- slider
- Storage
- stylesheet
- swf
- swfdetect
- swfstore
- tabview
- treeview
- uploader
- yahoo
- yuiloader

```
/**
 * The YAHOO object is the single global object used by YUI Library.  It
 * contains utility function for setting up namespaces, inheritance, and
 * logging.  YAHOO.util, YAHOO.widget, and YAHOO.example are namespaces
 * created automatically for and used by the library.
 * @module yahoo
 * @title  YAHOO Global
 */

/**
 * YAHOO_config is not included as part of the library.  Instead it is an
 * object that can be defined by the implementer immediately before
 * including the YUI library.  The properties included in this object
 * will be used to configure global properties needed as soon as the
 * library begins to load.
 * @class YAHOO_config
 * @static
 */

/**
 * A reference to a function that will be executed every time a YAHOO module
 * is loaded.  As parameter, this function will receive the version
 * information for the module. See <a href="YAHOO.env.html#getVersion">
 * YAHOO.env.getVersion</a> for the description of the version data structure.
 * @property listener
 * @type Function
 * @static
 * @default undefined
 */

/**
 * Set to true if the library will be dynamically loaded after window.onload.
 * Defaults to false
 * @property injecting
 * @type boolean
 * @static
 * @default undefined
 */

/**
 * Instructs the yuiloader component to dynamically load yui components and
 * their dependencies.  See the yuiloader documentation for more information
 * about dynamic loading
 * @property load
 * @static
 * @default undefined
 * @see yuiloader
 */

/**
 * Forces the use of the supplied locale where applicable in the library
 * @property locale
 * @type string
 * @static
 * @default undefined
 */

if (typeof YAHOO == "undefined" || !YAHOO) {
    /**
     * The YAHOO global namespace object.  If YAHOO is already defined, the
     * existing YAHOO object will not be overwritten so that defined
     * namespaces are preserved.
     * @class YAHOO
     * @static
     */
    var YAHOO = {};
}

/**
 * Returns the namespace specified and creates it if it doesn't exist
 * <pre>
 * YAHOO.namespace("property.package");
 * YAHOO.namespace("YAHOO.property.package");
 * </pre>
 * Either of the above would create YAHOO.property, then
 * YAHOO.property.package
 *
 * Be careful when naming packages. Reserved words may work in some browsers
 * and not others. For instance, the following will fail in Safari:
 * <pre>
```

yuitest

Classes

YAHOO
YAHOO.env
YAHOO.env.ua
YAHOO.lang
YAHOO_config

Files

Env.js
Lang.js
YAHOO.js

```
 * YAHOO.namespace("really.long.nested.namespace");
 * </pre>
 * This fails because "long" is a future reserved word in ECMAScript
 *
 * For implementation code that uses YUI, do not create your components
 * in the namespaces defined by YUI (
 * <code>YAHOO.util</code>,
 * <code>YAHOO.widget</code>,
 * <code>YAHOO.lang</code>,
 * <code>YAHOO.tool</code>,
 * <code>YAHOO.example</code>,
 * <code>YAHOO.env</code>) -- create your own namespace (e.g., 'companyname').
 *
 * @method namespace
 * @static
 * @param  {String*} arguments 1-n namespaces to create
 * @return {Object}  A reference to the last namespace object created
 */
YAHOO.namespace = function() {
    var a=arguments, o=null, i, j, d;
    for (i=0; i<a.length; i=i+1) {
        d=(""+a[i]).split(".");
        o=YAHOO;

        // YAHOO is implied, so it is ignored if it is included
        for (j=(d[0] == "YAHOO") ? 1 : 0; j<d.length; j=j+1) {
            o[d[j]]=o[d[j]] || {};
            o=o[d[j]];
        }
    }

    return o;
};

/**
 * Uses YAHOO.widget.Logger to output a log message, if the widget is
 * available.
 * Note: LogReader adds the message, category, and source to the DOM as HTML.
 *
 * @method log
 * @static
 * @param  {HTML}  msg  The message to log.
 * @param  {HTML}  cat  The log category for the message.  Default
 *                      categories are "info", "warn", "error", time".
 *                      Custom categories can be used as well. (opt)
 * @param  {HTML}  src  The source of the the message (opt)
 * @return {Boolean}    True if the log operation was successful.
 */
YAHOO.log = function(msg, cat, src) {
    var l=YAHOO.widget.Logger;
    if(l && l.log) {
        return l.log(msg, cat, src);
    } else {
        return false;
    }
};

/**
 * Registers a module with the YAHOO object
 * @method register
 * @static
 * @param {String}   name    the name of the module (event, slider, etc)
 * @param {Function} mainClass a reference to class in the module.  This
 *                             class will be tagged with the version info
 *                             so that it will be possible to identify the
 *                             version that is in use when multiple versions
 *                             have loaded
 * @param {Object}   data      metadata object for the module.  Currently it
 *                             is expected to contain a "version" property
 *                             and a "build" property at minimum.
 */
YAHOO.register = function(name, mainClass, data) {
    var mods = YAHOO.env.modules, m, v, b, ls, i;

    if (!mods[name]) {
        mods[name] = {
            versions:[],
            builds:[]
        };
    }

    m  = mods[name];
    v  = data.version;
    b  = data.build;
    ls = YAHOO.env.listeners;

    m.name = name;
    m.version = v;
    m.build = b;
    m.versions.push(v);
    m.builds.push(b);
    m.mainClass = mainClass;
```

```
            // fire the module load listeners
            for (i=0;i<ls.length;i=i+1) {
                ls[i](m);
            }
            // label the main class
            if (mainClass) {
                mainClass.VERSION = v;
                mainClass.BUILD = b;
            } else {
                YAHOO.log("mainClass is undefined for module " + name, "warn");
            }
        };
```

Copyright © 2011 Yahoo! Inc. All rights reserved.